Towards a closer integration between UMd MSEE curriculum and GSFC systems engineering education requirements.

Bernard Frankpitt, AIMS inc., Mark Austin, ISR, UMd.

November 12, 2003

This work was performed for GSFC, NASA under sub-contract to the University of Maryland, College Park.
Contract # N00173-01-C-2004

Contents:

The purpose of this contract.

- Contract arose from a partnership between NASA Goddard and ISR, UMd. under the auspices of a NSF Systems engineering education program.
- Purpose: To better integrate academic system engineering programs with the requirements of industry
- “Small” subcontract to AIMS to produce material illustrating how ideas from the academic curriculum can be better combined with the in-house education programs at NASA.

Approach

Develop material that would be suitable for short course presentations with the following goals:

- Illustrate point of view that UMd System engineering program takes to system modeling
- Look at the direction in which UMd sees this work going
- Examine how to link these ideas with NASA's System engineering methodology, particularly with regard to the early stages of the NASA system engineering process.

Deliverables:

- Report outlining material for approximately two short courses (We break the material in the report into two lectures)
- A presentation about the material

Summary of results:

Report provides illustrative material that covers three aspects of system modeling:

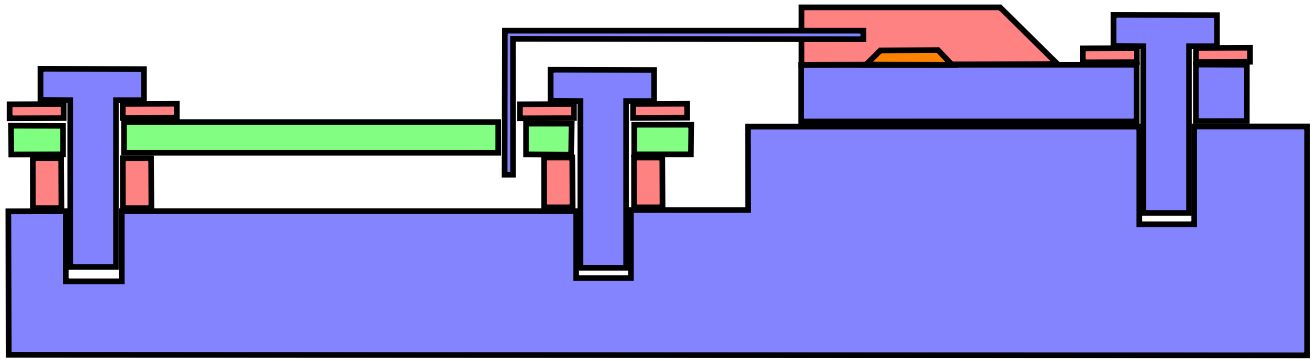
- Introduction of foundational modeling techniques.
- Show how the foundational techniques support new research into system engineering tools.
- Application of system modeling ideas to significant systems engineering issues at NASA.

Foundational Modeling Technique:

Object Oriented Methodology:

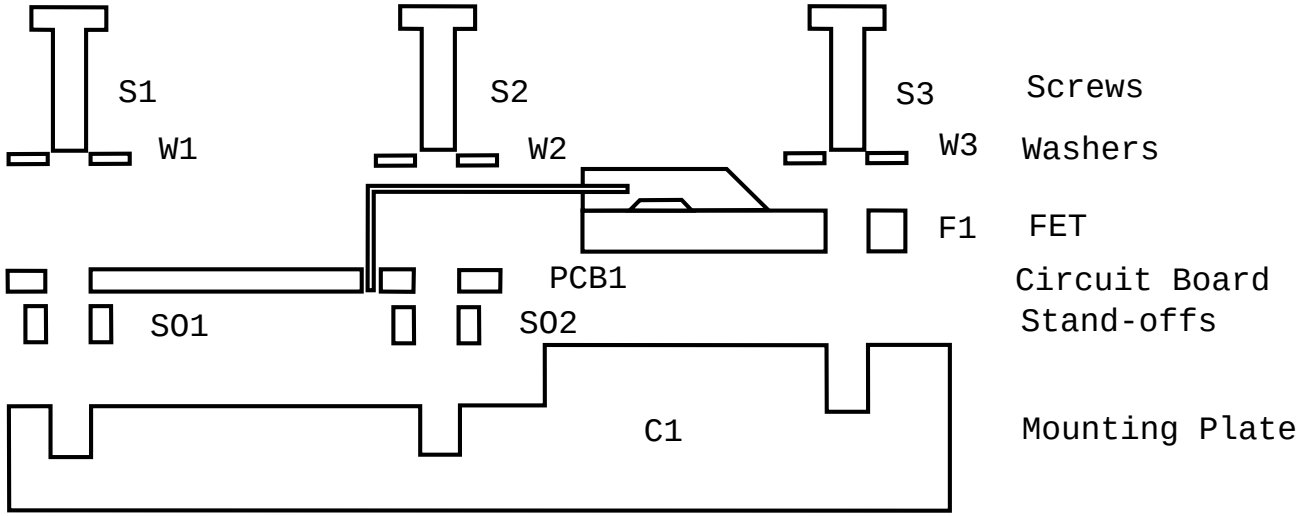
- Compact set of general purpose concepts for developing systems models.
- Develop the abstract methodology from concrete examples.
- Avoid the details of standards body specifications.
- The approach that we take is similar to Oliver, et al.

A simple example



A solid state amplifier mounted on a heatsink.

Assembly View

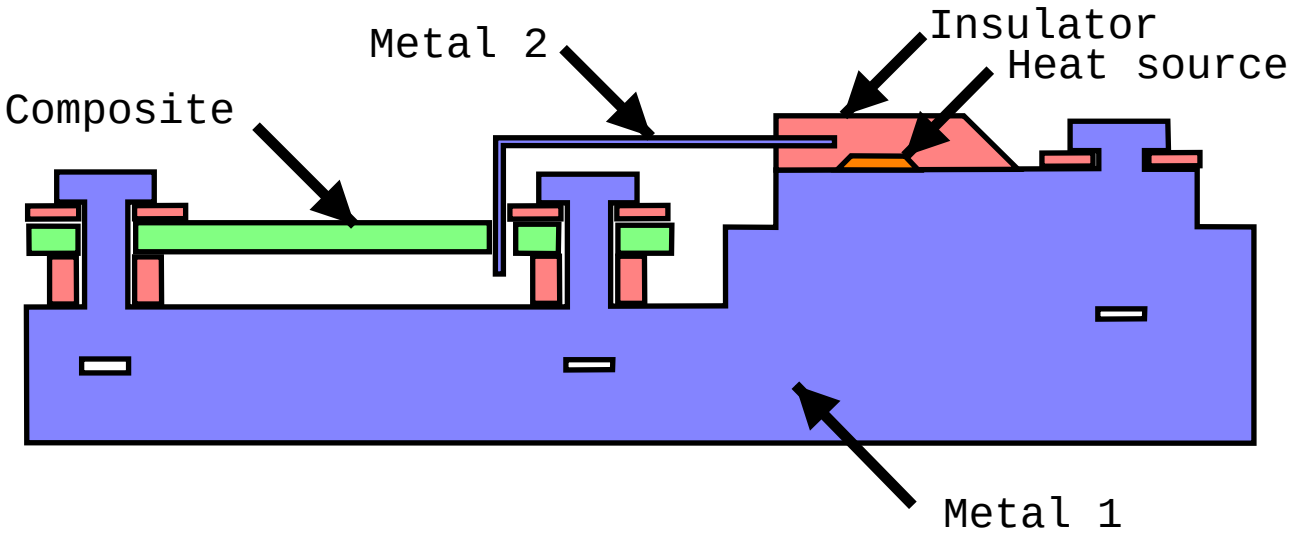


Classes: Screws, Washers, Standoffs, Heatsink, etc.

Class relationships constrain the assembly

"Each screw has a washer, a standoff and a threaded hole"

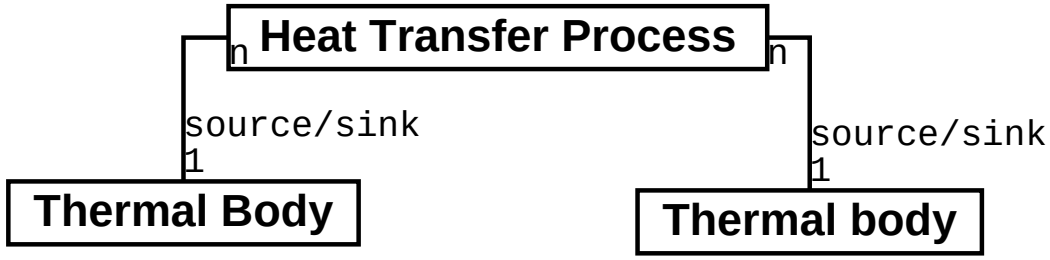
Thermal View



Classes: Homogeneous thermal bodies, Heat sources, Interfaces

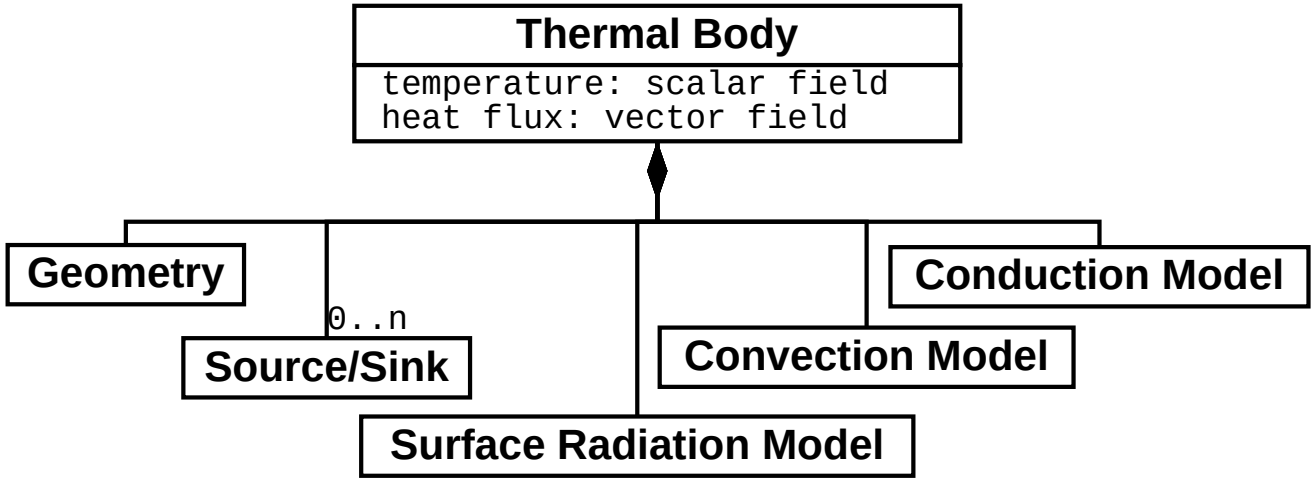
Class relationships specify PDE that determines temperature distribution

Thermal model details 1:



A **Thermal Process** controls heat transfer between two **Thermal Bodies**

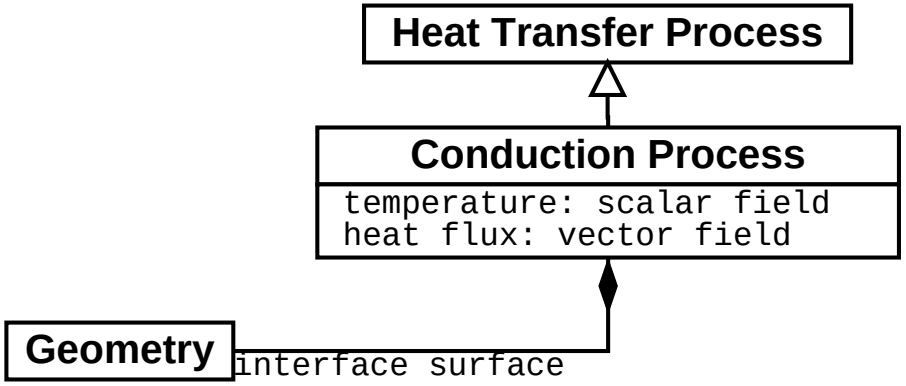
Thermal model details 2:



A **Thermal Body** has:

- A **Geometry Model**
- A set of **Sources** and **Sinks**
- etc.

Thermal model details 3:



A **Conduction Process** is a **Heat Transfer Process**. It has a **Geometry Model** that describes the interface surface, and heat flux and temperature fields defined on the interface surface.

Issues with object oriented system modeling

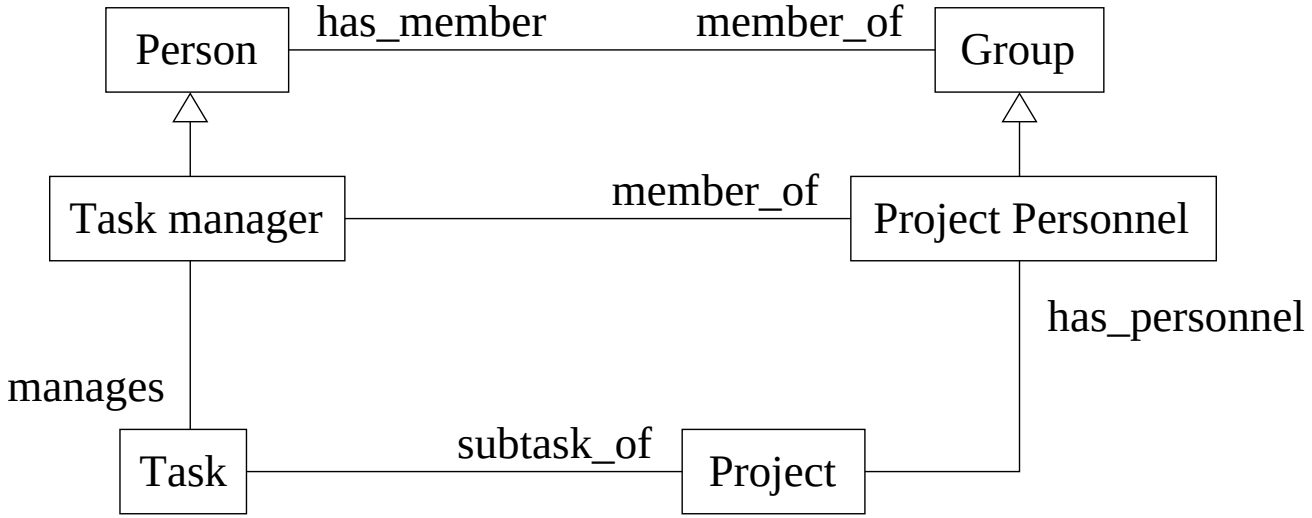
- Basis of standards such as STEP and OMG UML.
- Firm theoretical basis for future work.
- Separates system engineering and domain-specific concerns.
- Supports dynamic, heterogeneous, scalable models.

Web Ontologies

1. Object Oriented Models define both graphical and textual syntaxes
2. Web Ontologies develop the textual syntax of object oriented modeling in the context of WWW protocols.
   - Objects are labeled with URIs
   - Object and class relationships are described by RDF documents

Promised Result: A queryable distributed framework for system engineering models

Personnel model



- Implemented on personnel and project management databases.
- Describes data relationships within the database.

Personnel model (RDF fragment 1)

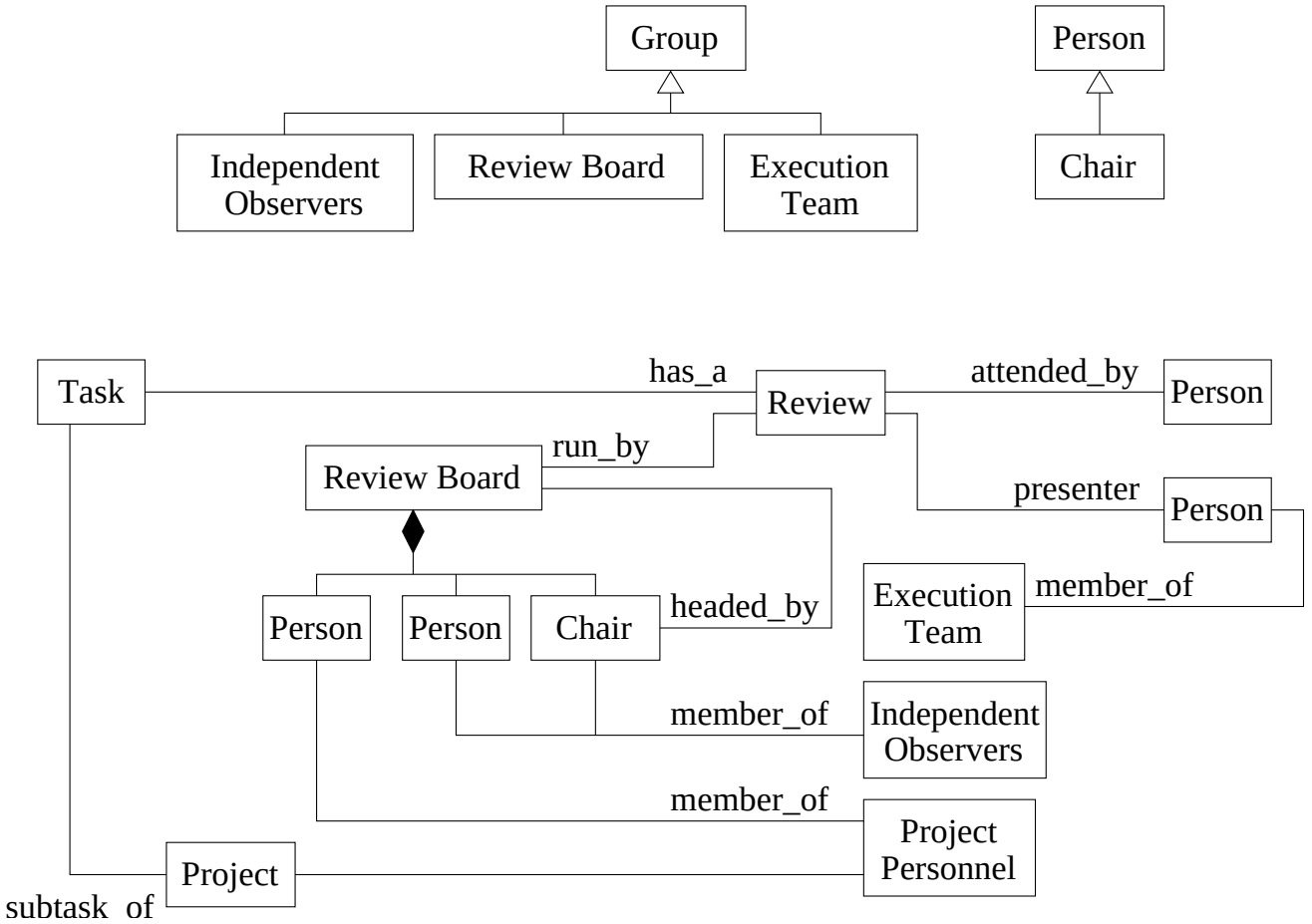
```
<rdf:RDF
    xmlns     ="http://aims-sys/WebOnt/personnel#"
    xmlns:owl ="http://www.w3.org/2002/07/owl#"
    xmlns:rdf ="http://.../1999/22-rdf-syntax-ns#"
    xmlns:rdfs="http://.../2000/01/rdf-schema#">

<owl:Class rdf:id="Person"/>
<owl:Class rdf:id="Group"/>

<owl:ObjectProperty rdf:ID="has_member">
  <rdfs:domain rdf:resource="#Group"/>
  <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>
</rdf:RDF>
```

Personnel model (RDF fragment 1)

```
<rdf:RDF
    xmlns     ="http://aims-sys/WebOnt/personnel#"
    xmlns:rdf ="http://.../1999/22-rdf-syntax-ns#">

<Person rdf:id="Bill"/>
<Person rdf:id="Alice"/>
<Person rdf:id="Roderick"/>
<Person rdf:id="Pamela"/>
<Person rdf:id="Terry"/>
<Person rdf:id="Wendy"/>

...

</rdf:RDF>
```

- Content and semantic structure separate from implementation.
- Namespaces provide modularity, and versioning.
- Ad-hoc organization is scalable.

Task Review


Group
Person
Independent Observers
Review Board
Execution Team
Chair
Task
has_a
Review
attended_by
Person
run_by
Review Board
presenter
Person
Person
Person
Chair
headed_by
Execution Team
member_of
member_of
Independent Observers
member_of
Project Personnel
Project
subtask of

```
<!DOCTYPE owl [
  <!ENTITY pers  "http://aims/WebOnt/personnel#"> ]>

<rdf:RDF
    xmlns:    ="http://aims/WebOnt/project#"
    xmlns:pers="http://aims/WebOnt/personnel#"
    xmlns:owl ="http://.../2002/ owl#"
    xmlns:rdf ="http://.../1999/22-rdf-syntax-ns#"
    xmlns:rdfs="http://.../2000/rdf-schema#">

<owl:Class rdf:id="Project"/>
<owl:Class rfd:id="Task"/>
<owl:Class rfd:id="ProjectPersonnel">
  <rdfs:subclassOf rdf:resource="&pers;Group"/>
</owl:Class>
```

```
<owl:ObjectProperty rdf:ID="subtask_of">
  <rdfs:domain rdf:resource="#Task"/>
  <rdfs:range rdf:resource="#Project"/>
</owl:ObjectProperty>
</rdf:RDF>
```

Implications for Systems Engineering tools

1. Ontologies are sets of propositions in a finite logic
2. Automatic theorem provers provide machinery to
   - Check models for logical consistency
   - Execute queries predicated on relations between entities

Example: An automated tool, integrated to existing data sources, that supports the management of project reviews.

Requirements and requirements processing

Outline:

- What is a requirement
- What is a requirements Document
- Early-phase requirements analysis: An example
- Detailed model of requirements processing

What is a requirement?

A requirement is best understood in the context of a procurement process:

- Customer presents producer with a contract opportunity.
- The two agree on requirements and verification procedure.
- Producer develops the product, and verifies requirements.
- Customer validates design by testing product in desired role .

Requirements regulate the design process at many levels of formality

Requirements Example:

- **Technician:** “Get me a 5mm m3 pan-head screw.”
- **Assistant:** “OK.”

Assistant fetches the screw. Technician unsuccessfully tries the screw in a hole

- **Technician:** “Oh Bother! its the wrong size.”
- **Assistant:** “It came from a new packet”
- **Technician:** “Well, lets try an 1/8th inch”

Requirements Example (cont):

- **Requirement:** A 5mm m3 pan-head screw.
- **Verification Procedure:** Check label on packet
- **Validation Procedure:** Try screw in hole

Note, successful verification does not imply successful validation, but it does shift the blame from the producer to the consumer.

The Consumer's Risk

Risk: The agreed verification procedure may pass, yet the validation may fail.

Mitigation: Specify a verification procedure that is as close to the validation procedure as possible.

Cost: A comprehensive verification procedure adds cost to the product.

The Producer's Risk

Risk: The requirements and verification procedure specify an empty design space.

Mitigation: Do advanced research and prototyping before accepting the contract.

Cost: The producer has to absorb the cost of the advanced research.

Requirements Processing and NASA's program lifecycle

Traditionally: Requirements are dealt with in formal documents or requirements tracking systems during Phase B

We propose: To loosen the definition of requirements and a requirements document to include less formal design contracts and requirements documentation that occur throughout the program lifecycle.

What are requirements in this context?

In design by successive refinement, each design iteration

- Starts with a system model, and a set of requirements.
- Elaborates and analyzes the system model to validate initial requirements.
- Generates requirements for following iterations.

Requirements are design contracts between groups of engineers and scientists working on distinct tasks in the program lifecycle.

Producers and Consumers are task teams working on distinct tasks.

Requirements negotiation occurs during design reviews.

What is a requirements document in this context?

The system's requirements document encompasses all system documentation that supports the tracking of requirements:

- Back to the assumptions and decisions that generated them.
- Forward to the point where they are validated in the design.

Requirements tracking and analysis has to support many styles of requirements documentation.

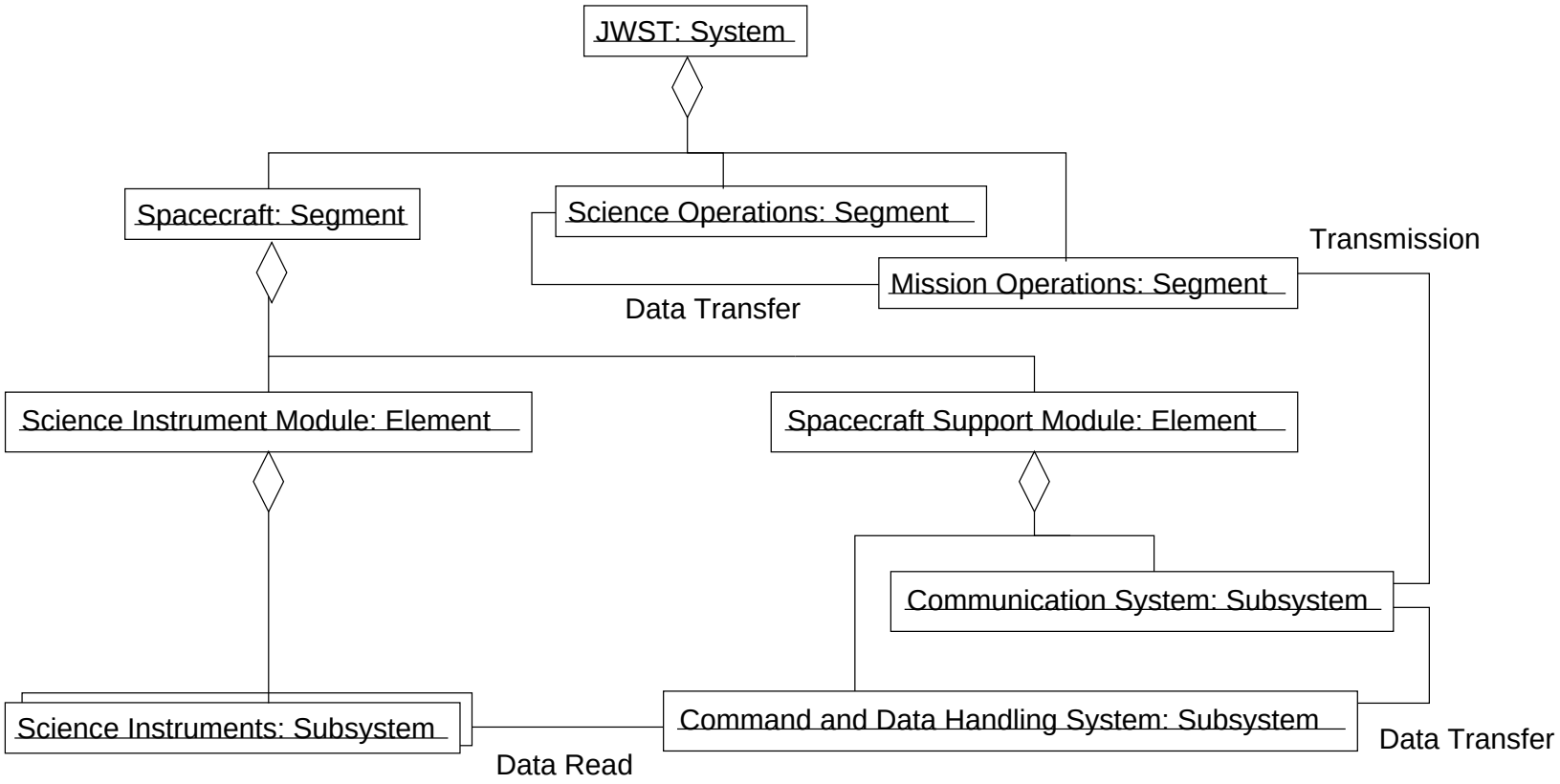
JWST Example:

NGST Data Volume and Communications Study

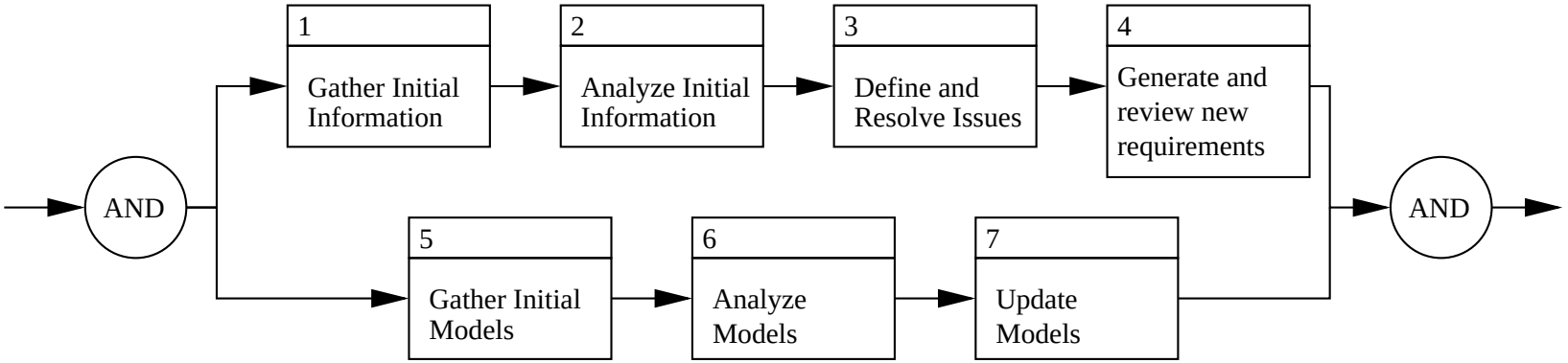
Study Objective: Develop requirements for data storage and link capacity for JWST.

Our Interest: An example of requirements processing and the development of a “Requirements Document” at an early phase of the NASA Program Life Cycle

System model for Data Comms Study



Information Assessment process



Requirements processing in the Data Comms survey is an example of an “Information Assessment Process” [Oliver et al.]

Initial Information (Boxes 1 & 5)

Initial model

- Yardstick mission: Conceptual mission design
- Design Reference Mission: Computational observational model
- Solar and Galactic radiation models

Original Requirements (From Yardstick Mission study)

- Science requirement: Acceptable rate of data loss.
- Basic functional requirement: Ability to skip a dump.
- Contingency requirement

Initial Information (cont.)

Design Requirements (Prior design constraints)

- X-Band/S-Band communications
- High-Gain reflector antenna
- S-Band omni-directional antenna
- low-level Communication protocols

Reference Requirement

- CCSDS/CFDP application protocol

Information Integration (Boxes 2, 3 & 6)

This is the major work of the study:

1. Determine study's scope – how much detail to consider.
2. Develop a set of initial requirements and models that are consistent with scope
3. Develop system design and models to:
   - Validate initial requirements
   - Generate developed requirements consistent with scope

Developed Information (Boxes 4 & 7)

Present the study's report as a consistent set of models and developed requirements.

The report becomes part of the system's "**Requirements Document**":

- Requirements that originate in the report are traced to the study's analysis of initial information and developed models.
- The study's original requirements are validated by the model analysis contained in the report.

Conclusions

Long term goals for our educational approach to systems modeling:

- Introduce simple, abstract modeling techniques that support the extraction of system engineering information from a wide variety of engineering models.
- Develop modeling languages that integrate well with modern ideas in distributed information processing, and support the development of system engineering tools that can handle heterogeneous, distributed sources of system engineering information.
- Reexamine classical system engineering problems, such as requirements analysis, to separate the fundamental abstract ideas from specific implementations. The result will be more

flexible and effective implementation of system engineering processes within the program lifecycle.